МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет информационных технологий

Кафедра параллельных вычислений

ОТЧЕТ

О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ

«ОПРЕДЕЛЕНИЕ ВРЕМЕНИ РАБОТЫ ПРИКЛАДНЫХ ПРОГРАММ»

студента 2 курса, 23202 группы

Пятанова Михаила Юрьевича

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:
Кандидат технических наук
Владислав Александрович
Перепёлкин

Новосибирск 2024

СОДЕРЖАНИЕ

ЦЕЛЬ

1. Изучение методики измерения времени работы подпрограммы.

2. Изучение приемов повышения точности измерения времени работы подпрограммы.

3. Изучение способов измерения времени работы подпрограммы.

4. Измерение времени работы подпрограммы в прикладной программе.

ЗАДАНИЕ

1. Написать программу на языке C или C++, которая реализует выбранный алгоритм из задания.

2. Проверить правильность работы программы на нескольких тестовых наборах входных данных.

3. Выбрать значение параметра N таким, чтобы время работы программы было порядка 15 секунд.

4. По приведенной методике определить время работы подпрограммы тестовой программы с относительной погрешностью не более 1%.

5. Составить отчет по лабораторной работе.

ОПИСАНИЕ РАБОТЫ

Перед началом выполнения лабораторной работы, на ПК была установлена программа VirtualBox 7.0.20 и создана виртуальная машина на операционной системе Ubuntu 24.04.1 LTS, с 8 ГБ. оперативной памяти и 2-х ядерным (4-х поточном) процессоре. Компилятором C/C++ был выбран gcc 13.2.0.

В ходе выполнения лабораторной работы был выбран первый вариант задания:

Алгоритм вычисления числа Пи с помощью разложения в ряд (ряд Грегори-Лейбница) по формуле Лейбница N первых членов ряда:

$$\pi = 4 \sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1}$$

Программа, реализующая выбранный алгоритм, была написана на языке программирования C++ (прил. 1). Правильность работы программы была проверена. Алгоритм отработал корректно: уже при N = 100000 алгоритм выводит верные первые 6 цифр числа π. Ниже прилагается таблица тестовых данных и вывода программы:

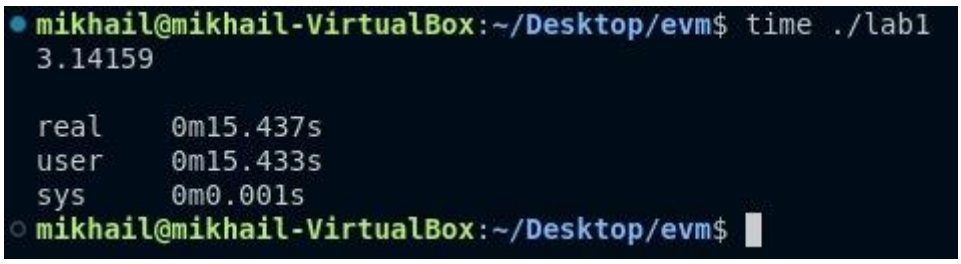
Ввод	Вывод
N = 10	π = 3.23232
N = 100	π = 3.15149
N = 1000	π = 3.14259
N = 10000	π = 3.14169
N = 100000	π = 3.14159

В свою очередь методикой определения времени работы программы был выбран метод 4.1 “Утилита time”.

Утилита time измеряет время работы приложения во многих конфигурациях ОС GNU Linux/UNIX. Утилита time выдаёт следующие временные характеристики работы программы: real – общее время работы программы согласно системному таймеру, user – время, которое работал пользовательский процесс (кроме времени работы других процессов) и sys – время, затраченное на выполнение системных вызовов программы.

Данный выбор основывается на удобстве: готовая утилита, не требуется вносить изменения в программу; точность определяется точностью системного таймера и точностью измерения времени работы процесса. Точность системного таймера в ОС GNU Linux/UNIX высока и составляет 1 нс ($1 \cdot 10^{-9}$ с).

Далее путем перебора входных данных было выбрано значение параметра N такое, чтобы время работы входной программы было порядка 15 с.: N = 3000000000. Ниже прилагается снимок экрана:



Таким образом, относительная погрешность была получена по формуле и равна $\frac{1{,}00*10^{-3}c.}{15{,}44\ c.} * 100\% = 6{,}48 * 10^{-3}\ \%$.

Была сделана попытка увеличить точность вычислений путем различных приемов уменьшения влияния посторонних факторов на время выполнения прикладной программы. Ниже представлены эти приемы:

Многократное измерение и сброс буфера отложенной записи на диск.1-3 измерения сделаны до сброса буфера, 4-5 сделаны после (прил. 2). Также во всех измерениях стадии инициализации и завершения были исключены (прил. 1). Посторонние процессы были завершены (настолько, насколько это возможно).

Номер измерения	Полученное время (s)
1	real 15.401 user 15.392 sys 0.006
2	real 15.480 user 15.475 sys 0.003
3	real 15.449 user 15.447 sys 0.001
4	real 15.443 user 15.434 sys 0.006
5	real 15.382 user 15.374 sys 0.006

Из всех полученных измерений наиболее точным будет минимальное. Это связано с тем, что сторонние факторы вносят различный вклад в измеряемый интервал при каждом запуске. Время же работы самой программы остаётся неизменным.

Новая относительная погрешность была получена по формуле и равна $\frac{1{,}00*10^{-3}c.}{15{,}38\ c.} * 100\% = 6{,}5 * 10^{-3}\ \%$.

ЗАКЛЮЧЕНИЕ

Получен опыт работы с Ubuntu и виртуальной машиной, shell.

Изучены методики измерения времени работы подпрограммы, приемы повышения точности измерения времени работы подпрограммы; способы измерения времени работы подпрограммы. Измерено время работы подпрограммы в прикладной программе.

Посчитана относительная погрешность до и после оптимизации кода. Приемы повышения точности измерения показали свою эффективность, улучшив результат с 15.437 с. до 15.382 с.

Приложение 1.

Команда для компиляции:
g++ lab1.cpp

Код до оптимизации

```cpp
#include <iostream>

long double GetSign(size_t n) {
    return n % 2 == 0 ? 1.0 : -1.0;
}

long double GetPiNumberWithNAccuracy(size_t N) {
    long double res = 0;
    for (size_t i = 0; i <= N; i++) {
        res += (GetSign(i) / (2*i + 1));
    }
    return 4 * res;
}

int main() {
    int n = 0;
    std::cin >> n;
    for (int i = 0; i <= n; i++){
        std::cout << GetPiNumberWithNAccuracy(i) << "\n";
    }
    return 0;
}
```

Код после оптимизации

```cpp
#include <iostream>

long double GetSign(size_t n) {
    return n % 2 == 0 ? 1.0 : -1.0;
}

long double GetPiNumberWithNAccuracy(size_t N) {
    long double res = 0;
    for (size_t i = 0; i <= N; i++) {
        res += (GetSign(i) / (2*i + 1));
    }
    return 4 * res;
}

int main() {
    int n = 3000000000;
    //std::cin >> n;
    std::cout << GetPiNumberWithNAccuracy(n) << "\n";
    return 0;
}
```

Приложение 2.